Victorian 6502 User Group Newsletter

KAOS

For People Who Have Got Smart

OSI SYM KIM AIM APPLE UK101 ORANGE

Registered by Australia Post
Publication No. VBG4212

ADDRESS ALL CORRESPONDENCE TO 10 FORBES ST., ESSENDON, VIC. 3040

Vol.2 No.12 September 1982

This issue of KAOS marks our second birthday, (no cards please, cash preferred). We are all looking forward to another delicious birthday cake - hint hint, nudge nudge!!

When David, Jeff and I started this group two years ago, we were thinking in terms of 15 to 20 members and that we would be able to produce the newsletter on unattended photocopiers. We bought a second-hand copier when our membership reached 100 and with a fair amount of maintenance and prayer, it has served us well. Now, because we are producing 450 copies of each issue and the copier was not designed for this volume of work, the quality of the copying deteriorates towards the end of the run. If you are unlucky enough to get one of these copies, you will no doubt have noticed this lack of quality. Incidently, it takes 8½ hours to copy the newsletter, or it would if nothing went wrong.

All this has lead us to another giant step forward - TRA RAH! - from this issue, the newsletter will be off-set printed. This will effect you in two ways, hopefully the newsletter will be easier to read, and as the newsletter will take a week longer to produce, the closing date for articles etc. will be earlier. To make this easier for you, in future we will include the closing date for the next issue on this page. Incidently, for those members who are not sure when their newsletter will arrive, barring postal strikes etc., the newsletter is posted on the Monday before the meeting which is on the last Sunday of the month. Hands up anyone who understands that!

The next meeting is on Sunday 26th September 1982 at 2pm at the Essendon Primary School which is on the corner of Raleigh and Nicholson Streets, Essendon.

The closing date for items for the next issue is 15th October.

****

INDEX

HOW TO PUT A BASIC PROGRAM AT ANY ADDRESS

With CMOS memory available with battery back-up you may want to put your most used BASIC programs in it or put them in EPROM as I have done. This is how to do it.

A BASIC program has to be written at the address it is RUN at or LOADed there from tape. It can't be moved there with the Extended monitor because each BASIC line is stored in memory with the address of the next line. In this example it is assumed 8k of Ram at $0000 to $1FFF and another 8k at $8000 to $9FFF

The following are the BASIC work space pointers that have to be altered

Pointer address		cold start value	new value for loading	new value for editing or running	new value running only
$0079	text start LO	01	01	01	01
$007A	text start HI	03	80	80	80
$007B	text end LO	03	03	see note	see note
$007C	text end HI	03	80	A	B
$0085	memory end LO	00	00	00	see note
$0086	memory end HI	20	A0	A0	C

Note A. This has to contain the address of the next location after the three BASIC nulls that are at the end of every basic program. It is updated automatically. If you want to return to a program and edit it then this has to be noted before leaving, then restored when returning.

Note B. This can be the end of the new program or the end of a program that is in $0300 to $2000

Note C. This is the end of memory and must be the same memory area as in $007B-$007C.

Procedure:- Do a cold start. Break M. Change contents of memory at $007A, $007C and $0086 as above. Put 3 nulls at the start of the new memory. ie 00 at $8000,$8001 and $8002.
Type in your BASIC program or LOAD it from tape. (This will be loaded at $8000)
When it's working correctly do a Cold start.

Another program can now be loaded in normal memory. To use the program at $8000 POKE122,128 and RUN. (122=$7A 128=$80).
To go back to normal POKE122,3.

I have a program in EPROM that uses Ext. Monitor subroutines, this allows me to cold or warm start BASIC at any RAM address. Eg *W0300,2090 will search for the 3 nulls at the end of the BASIC program, then poke the correct values into $0079 to $0086.

John Whitehead

****

FIBREGLASS SUPERBOARD CASES. Attractive off-white fibreglass cases with a sheet metal base, and these features:
*Secure seven bolt mount for the Superboard & other printed circuit boards
*Mains socket for monitor, & RCA Video socket mounted on the case's rear.
*Mounting place for cassette player.
*Switch-plate allowing for mains switches, power indicators etc.
*Plenty of room for a power supply & expansion boards.
$55.00 + T for 1-9, $50.00 + T for 10+. Gavin Eakins

****

5 Volt, 3 AMP Power Supply, suit SUPERBOARD. Cost $45, Sell $25
Gavin Eakins

INTO THE PEACH

Whilst trying to write, modify or investigate programmes for the PEACH, one comes to a stale mate with no knowledge of ROM etc. Here are a few locations, with many thanks to George, that I have found useful.

ZERO PAGE

9E	Output Suppress. 00=normal. 01=Suppress
277,377	Line Buffer
117,118	Suppress List Flag. FFFF=Enable. 0000=Suppress
0B4	File Protect Flag FF=Protected. 00=Not Protected.
1D,1F	Beginning of Basic
88	Trace Flag. 0=Off. 1=On.

ROM ROUTINES

A438	Warm Start	E6D4	LOAD Routine
A5BD	RUN Routine	EE3C	MOTOR Routine
BD6C	RENUM Routine	EEC5	Save to ACIA Routine
DCE6	MON Routine	EED7	RESET Start
DE70	EXEC Routine	F0D1	Screen Output
E39E	FILES Routine	F160	CLS Routine
E4BE	LIST Routine	F7A7	Bell Sound
E4FA	SAVE Routine	FAB0	Cold Start
E5AD	SAVEM Routine		

Please contact me if you want a fuller listing.

Erik Sundstrup (054)261921

Here is a short programme to show the power of the INPUT$ command. It is a screen dump routine. Draw the screen and make many changes with inserts and deletes etc. When finished hit "#" and see the screen reconstructed exactly as you made it. If you do not have a printer just RESET and start again.

```
10  CLOSE:SCREEN0:WIDTH80:CLS:CLEAR 2500:DIM A$(2500)
20  LINE(0,0)-(639,199),PSET,B
30  A$(X)=INPUT$(1):PRINTA$(X);
40  IF A$(X)="#"THEN A$(X)="":GOTO 50 ELSE X=X+1:GOTO 30
50  CLS:FOR T=0 TO X:PRINTA$(T);:NEXT T:OPEN"O",2,"LPT0:"
60  FOR G=0 TO 2000:IF PEEK(1024+G)< 32 THEN POKE 1024+G,32
70  PRINT#2,CHR$(PEEK(1024+G));:NEXTG:CLOSE:CLEAR300
```

Erik Sundstrup

****

TIPS
by Garnett Znidaric

A few tips to help out beginning hardware and software hackers.

- Flashing LEDs run on 5Vdc and can act as power indicators.
- Always work out a sample run so that you know whether the formula or your system is at fault.
- Label everything that comes apart so that re-assembly will be straight forward.
- In building up component boards etc. always draw out your wiring diagrams. Over the years (months) memories fade. Diagrams don't.
- Soldered leads are a hassle, and break easily. Make things plug together modules where possible.
- With 40 pin plugs (which catch easily in the carpet, use a spare cheap socket as the 'plug' by pressing the plug into it. 40¢ sockets are cheaper than $5 plugs.
- Always make a backup. I always try, but I still get caught out.
- Don't forget, often a dry solder joint is the cause of the problem and the hardest to find.
- Some fine solder joints can get hot enough to melt the solder.
- Rule number one is always start at the power supply.
- Enclosed systems can cook. Ensure your chips and regulators have enough ventilation.

Superboard

NEWSLETTER OF THE OHIO SUPERBOARD USER GROUP, 146 YORK STREET, NUNDAH, 4012.

ANIMATION

Cegmon owners can use the M/C block move facility to generate fast animation on the screen. Using this facility, you can swap any memory to or from the screen or any other memory. This program demonstrates:-

```
10 POKE 254,A:POKE 255,160:REM START ADDRESS OF BLOCK TO MOVE
20 POKE 249,255:POKE 250,167:REM END      "     "    "    "  "
30 POKE 228,20:POKE 229,208:REM NEW START OF BLOCK(SCREEN)
40 POKE 11,228:POKE 12,253:X=USR(X):REM CALL $FDE4
50 A=A+1:IF A=256 THEN A=0:REM INCREMENT START ADDRESS
60 GOTO 10
```

MEAN TRICKS DEPT by Ross Baker

Typing POKE 5,162 into a continuous running program will bomb it completely on a BREAK, W.
This next program prints up a dandy message when you try to save:-

```
10 POKE 544,34:POKE 545,2:FOR T=546 TO 638:READ A:POKE T,A
20 NEXT
30 DATA 160,0,169,32,153,0,208,153,0,209,153,0,210
40 DATA 153,0,211,200,208,241,160,0,185,72,2,201,255
50 DATA 208,3,76,0,0,153,5,210,200,76,55,2
60 DATA 29,32,82,101,99,111,114,100,105,110,103,32,111,102
70 DATA 32,116,104,105,115,32,32,32,32,32,32,32,32,32,32,32
80 DATA 32,112,114,111,103,114,97,109,32,105,115,32,80,82
90 DATA 79,72,73,66,73,84,69,68,32,29,255
```

A much simpler way would be to call the memory test routine with a couple of pokes, POKE 544,137:POKE 545,189, or the NEW routine, POKE 544,99:POKE 545,164 To be of maximum effect, the SAVE vectors would need to be placed early in the program and disguised with variables, additions, etc. An automatic start would also be needed. Still, BREAK,W,SAVE will always defeat.

A very sneaky one is to configure the ACIA to send break on data out, with POKE 61440,3:POKE 61440,113, again heavily disguised in the program. Everything seems to SAVE normally on the tape, however when you try to LOAD it later on, a nasty surprise awaits.

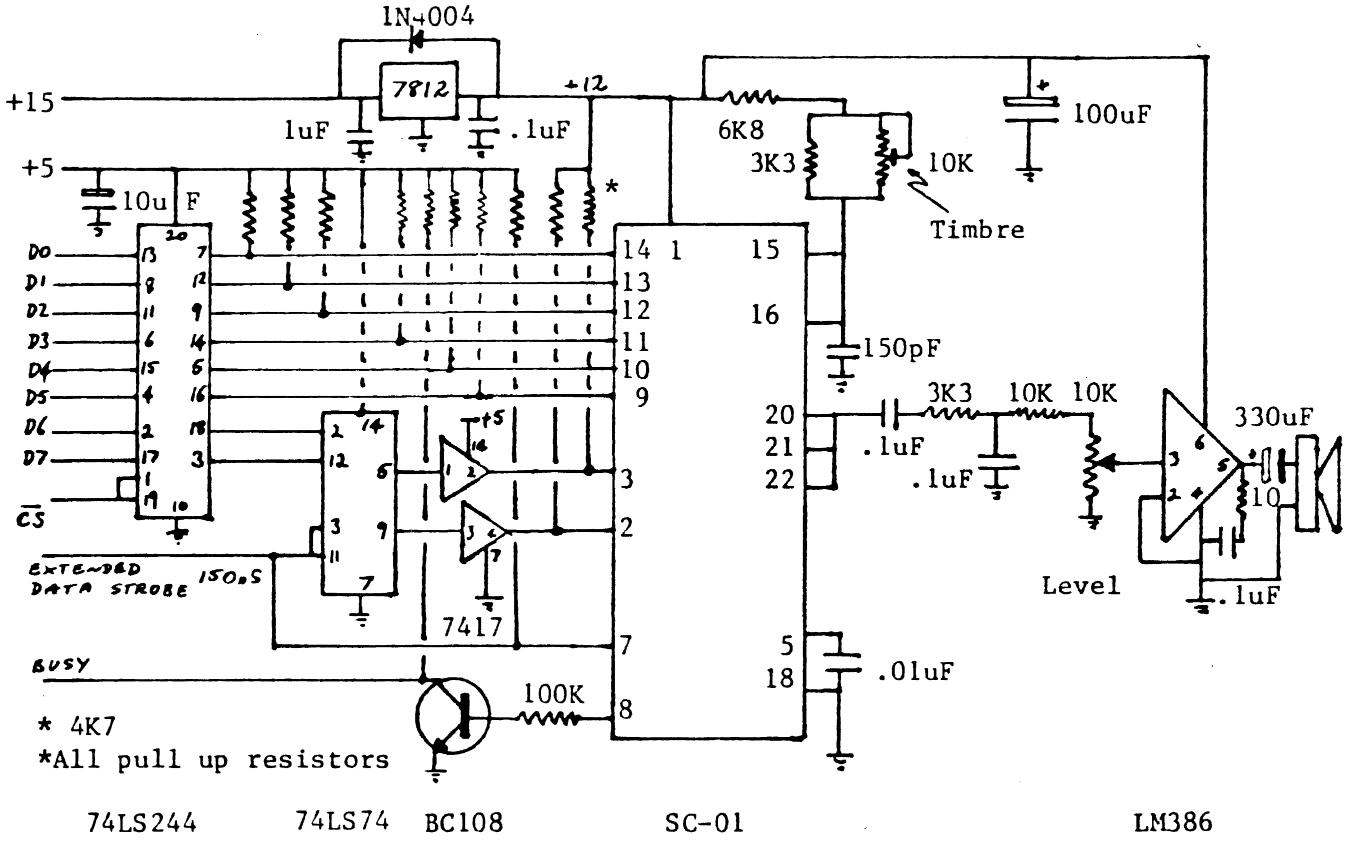
STRING BUG

Trevor Stephenson has reminded me that the OSI string bug can often be avoided by always dimensioning string arrays in a number of the sequence N*3+2 where N is any number. Trevor has found that this works for a number of Adventures.

CHARACTER SETS - SCIENTIFIC

In addition to the various character sets mentioned last month, a scientific set is now available. All normal keyboard characters, reduced in size to improve readability, are there but the gaming characters have all been deleted and replaced by subscripts, superscripts and various scientific symbols, including the Greek ones. The set can be ordered in a 2716 to replace the current character generator, or piggyback onto it. Alternately, the scientific set can come as one half of a 2732, with any of the sets mentioned in KAOS 2/11

Superboard



PHONETIC SPEECH SYNTHETIZER by Ray Richards

The circuit was built on a Dick Smith H5606 board. The SC-01 is also obtained from there and costs just under $70. Catalogue number is Z6818. Only a few other commonly available I.C.s are required to make up the circuit.

The synthetizer needs a parallel port to drive it, and a PIA or VIA would be ideal. Another use for a Tasker I/O board if you have one.
If you mount the PIA on the board, the LS244 is not necessary.
A serial to parallel converter can be made using a small board and several I.C.s, and plugged into the ACIA socket. This is also useful for driving a parallel input printer.

Several programs have been developed for the synthetizer. A simple one is used to assemble phonemes into possible words, and you can easily add to or modify these. When you have the sound of the word to your satisfaction, the decimal numbers of the phonemes used can be displayed for later incorporation in programs.

A practical use for the device was an alarm system to give an audible indication of the fault, followed by more information being printed out. AIRCONDITIONING FAILURE or TYPESETTER ERROR are spoken by the synthetizer. Specific instructions are then printed explaining the steps to take in rectifying the problem.

A program to assemble phonemes will be given in the October SUPERBOARD.

More information is available if you are interested in building the synthetizer. Write to Ray at 7/39 Ross Street, Parramatta, NSW 2150 or to this User Group for more information. Please enclose a 9"x4" SAE with any queries.

- -

All going well, the next User Group Meeting will be on September 26th. I will be contacting local members. 1pm, 3rd floor, Sherwood House, Toowong

Ed Richardson

The numbers the 6502 uses as instructions are usually referred to as opcodes (short for operation codes). The opcode provides 3 pieces of imformation needed by the CPU:

(a) which of the 56 instructions is to be performed
(b) which register if any is to be used
(c) which of the 13 addressing modes will apply.

Often the CPU will be required to perform some operation on memory. For this it needs to know what address is to be placed on the address bus. To describe a specific location absolutely requires 16 binary digits (2 bytes). If the opcode calls for this ABSOLUTE MODE of addressing the CPU will expect that the 2 bytes following the opcode will give the data for loading onto the address bus. The instructions

```
    LOAD THE ACCUMULATOR FROM LOCATION $000A    (A=PEEK(10))
    INCREMENT MEMORY LOCATION $B140
    STORE THE X REGISTER AT LOCATION $1000   (POKE 4096,X)
```

aqe examples of absolute mode instructions.

When the operation is to be carried out internally (between registers) or between the registers and the stack, there is no address needed. This addressing mode is known as INHERENT or IMPLIED mode. In this mode only the opcode is required. Examples of implied mode instructions are:

```
    TRANSFER THE X REGISTER TO ACCUMULATOR   (A=X)
    DECREMENT THE VALUE IN THE Y REGISTER   (Y=Y-1)
    SET THE DECIMAL FLAG   (D=1)
```

At times it is necessary to set the contents of some register to a known value eg LOAD THE ACCUMULATOR WITH $20   (A=32)
This is done with IMMEDIATE MODE instructions. In this mode the value in the byte following the opcode is to be loaded into the register specified by the opcode.

These 3 addressing modes and the one we are about to cover are the ones most commonly used in programming.

The fourth addressing mode is used exclusively by the branch instructions and is known as RELATIVE MODE. In relative addressing the byte following the opcode is used to calculate an address relative to the program counter. There are 2 concepts which must be understood before you can come to grips with relative addressing:

(1)the program counter is automatically incremented as each byte is loaded from the data bus (ie as each instruction is processed). Therefor by the time the instruction has been read the program counter is pointing at the next instruction.
(2)If the value of the byte is greater than $7F (127) then it is assumed to be a negative value expressed in 2's complement notation.

By using this convention it is possible to branch foreward or backward in a program.

The simplest way to show how to find 2's complement of a number is to do it. Take a number say $A7

in binary this is	10100111
reverse the digits	01011000
add 1	01011001
convert it back to hex	=-$59 (-89)

The maximum negative value that can be expressed in 2's complement is $80 (-128). Thus the effective range of the relative branch is +127 or -128. However, remember that the program counter has been advanced 2 bytes passed the branch opcode so the range becomes 2+127=+129 or 2-128=-126 relative to the branch.

So far we have only described machine language instructions in terms of words. While this makes the listing of a program very readable, it does tend to be a little tedious having to write it all out. Unfortunately too these phrases have no relationship to the computers binary language.

Most people would not be too happy with learning some 150 hexadecimal numbers just so they can "tell the computer what to do". What is needed then is something simpler that describes what is going on at machine level when we read it and which is capable of being easily converted to machine language. To achieve this mnemonics are used. Mnemonics are compressed descriptions of the opcodes. For example

LoaD Accumulator	=LDA
STore Accumulator	=STA
Jump to SubRoutine	=JSR
JuMP	=JMP
Transfer X register to Accumulator	=TXA
CoMPare	=CMP

this gives us a short but still relatively human friendly (readable) programming language.

Mnemonics also give us the ability to describe the addressing mode simply. For absolute mode we just give a four digit absolute address in Hex eg

```
     LDA $000A
     INC $B140   (INC=increment)
     STX $1000   (STX=store X register)
```

Inherent instructions have no address just the mnemonic. In immediate mode the data is prefixed by the # symbol. eg LDA #$20
Relative mode is easily identified because all the instructions are branches. eg

```
     BEQ $F7   Branch if EQual to zero
     BNE $40   Branch if Not Equal to zero
     BMI $05   Branch if MInus
```

When the time comes to actually put the opcodes into memory we can look up the numbers and do the conversion manually or use an assembler to do it all for us.

David Dodds

****

There were a number of errors in last month's article on mods to Disk Basic. After 17C0 the next four lines should read:

```
17C3  202E0F  JSR  0F2E  get address of second parameter
17C6  850A    STA  0A    swap A register and Y register
17C8  98      TYA        to get high byte in A
17C9  A40A    LOY  0A    and low byte in Y
```

Rodney Eisfelder

****

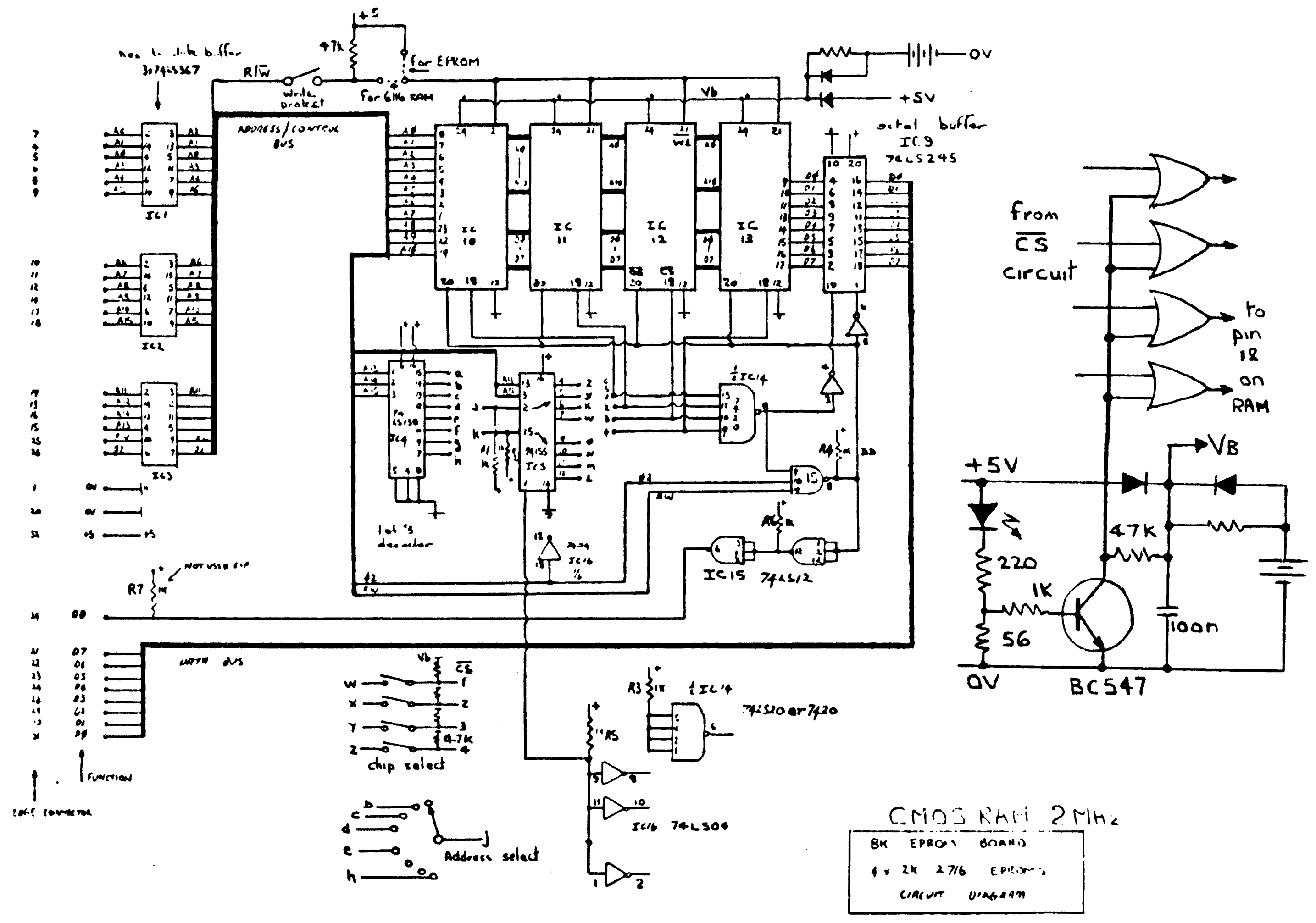
A NOTE FOR VIC 20 USERS

Of potential interest to any Vic 20 users, I recently developed a technique for burning BASIC programs into EPROM to run automatically on power-up. If any members are interested in the technique, I would be happy to supply details.

Mike Phillips

6116LP-3 CMOS RAM on a TASKER BOARD

by John Whitehead



The circuit above shows how I use 6116 CMOS RAM with battery back-up on a Tasker buss EPROM board. The 6116LP-3 switches to low power data retention state when $\overline{\text{CS}}$ is within 0.2 volts of Vcc and Vcc is above 2 volts. It then draws between 2 and 50 microamps instead of 4 to 70 milliamps. The current taken by 4 chips is too low to measure on my Fluke DMM.

The circuit I used does not contain any auto power down switching as this is not needed if the RAM is not being accessed at the time of power down, but I have shown a circuit for auto power down which could be useful if you have CMOS RAM at $0000 onwards.

To make the RAM work at 2MHz the $\overline{\text{CS}}$ has to go low when the address lines are set. To do this, connect pin 18 of IC10, 11, 12 and 13 to pins 13, 12, 10 and 9 of IC10. Also remove 02 from IC5 by connecting pin 14 of IC5 to 0V and pin 1 of IC5 to pin 9 of IC16. Connect $\overline{\text{OE}}$ to DD, that is pin 20 of IC10, 11, 12 and 13 to pin 8 of IC15. Connect $\overline{\text{WE}}$ to R/W via a SPST switch for write protection, that is pin 21 of all 6116's to a 47k resistor and to the switch, the other side of the resistor to +5V and the other side of the switch to pin 11 of IC15.

The 74LS241 buffer can't be used as this is only single direction. I have used a 74LS245 but this needs 9 tracks cut, 9 links and an extra inverter. The buffer can be by-passed by linking D0 to D0 etc.

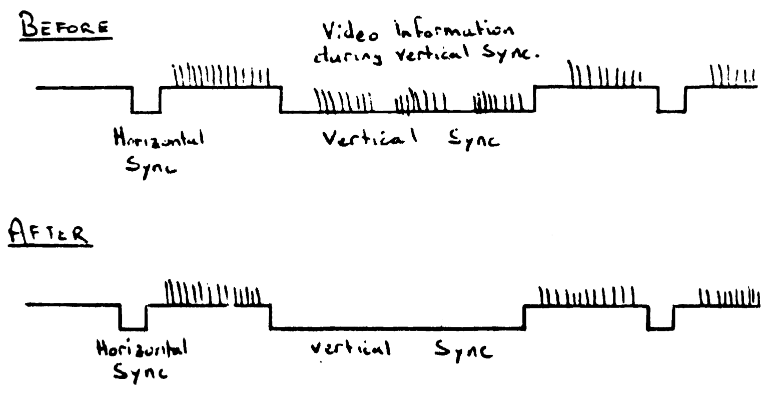
The battery I have used is a 3.6V Ni Cad with a 270 ohm charging resistor which charges at 800 microamps. A 2.4V battery could be used.

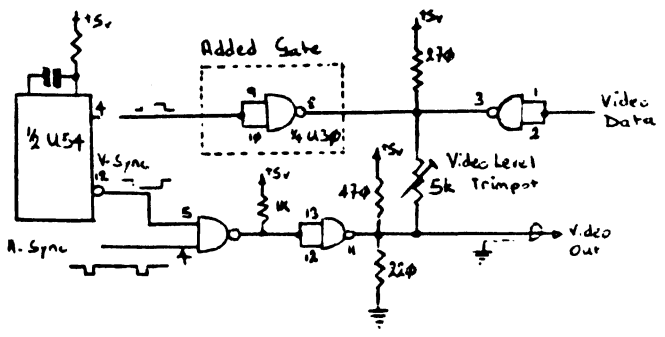
I use the same connections as above to pins 18 and 20 on 2716 EPROMs with pin 21 connected to +5V.

ANOTHER MOD FOR THE VIDEO BOARD

I have done a small mod to my TASAN Video board which may be of some interest Whilst tracking down a bent I.C. pin with a CRO, I was disturbed to discover 'video' appearing during the vertical sync pulse. Whilst this might not be a problem with most monitors, as an ex video technician I decided to cure it. The solution is to feed positive going vertical sync pulses from U54 pin 5 through the spare gate on U30, (input on pins 9, 10, output on pin 8) to pin 3 of U30, thus nobbling the video during vertical sync. Diagram below.

Mike Phillips





****

EXTENDED MONITOR SEARCH ROUTINES PART II.

In the April issue of the KAOS newsletter, John Whitehead published a patch for the Extended Monitor to enable searches for multiple occurrences of a search string. Unfortunately for users of disk systems, the mods were for the cassette version of the Extended Monitor, which is slightly different and resides in a different area of memory from the disk version.

The following patch has the same affect on the disk version of EM as John's had on the cassette version (except that the 'Z' command is not affected because there is no 'Z' command).

```
1ABA  4C3ELF  JMP  1F3E  (to patch)
1F3E  F007    BEQ  1F47
1F40  C94E    CMP  #4E   check for 'N'
1F42  F004    BEQ  1F48
1F44  4CBD1A  JMP  1ABD
1F47  60      RTS
1F48  A5CA    LDA  CA    search from this point on
1F4A  85CC    STA  CC
1F4C  A5CB    LDA  CB
1F4E  85CD    STA  CD
1F50  4CC31C  JMP  1CC3  search
```

These mods should be applied to tracks 10 and 11 on mini floppy systems (track 10 belongs at 1700-1EFF and track 11 belongs at 1F00 to 22FF, although only the first 60 bytes are used). Note well that track 11 sector 1 must not be more than 4 pages long or else the operating system will be clobbered when it gets loaded.

As an example of the use of this extension, the following example finds the first two occurrences of 'A9' (LDA immediate) in the Assembler.

```
:NA9>0200,0400(CR)
022E/A9  (LF)  first occurrence
022F/07  N     search again
0235/A9        second occurrence
```

Rodney Eisfelder

THE SPEAKER

A common mistake made by SYM owners is to assume that the white and gold disc next to the keyboard is just a beeper, capable of producing a single type of tone. It is, in fact, a small piezo-electric speaker driven from an I/O port via a transistor and is capable of producing a wide range of tones and noises.

The I/O port at $A402 is shared by the keyboard, display, and speaker. To select the speaker as the device to be used, use the instruction sequence:

```
    A9 0D     LDA #$0D
    20 A589   JSR CONFIG
```

The speaker will remain as the selected device until it is altered. Therefore, the above instructions would form part of the initialization routine.

The tone is produced by applying alternating ones and zeros to the I/O port with a short delay. ie.

```
    A9  08      LOOP    LDA  #$08
    8D  02A4            STA  PBDA
                        delay
    A9  06              LDA  #$06
    8D  02A4            STA  PBDA
                        delay
    4C  XXXX            JMP  LOOP
```

The length of the delay determines the frequency of the tone.

While this technique will allow you to play tunes with your SYM, you will soon realize that there are a few shortcomings. Firstly, there is not much volume. This will be a problem for those who wish to share their musical masterpieces with the rest of the household or maybe the neighbours as there is no socket allowing the connection of your 300W amplifier. The other problem is that it is not possible to play harmony as well as the melody. Could you imagine the 1812 Overture played as a one-note melody?

Both problems can be solved by ignoring the on-board speaker and connecting your own. There is a further advantage in using PB7 on an I/O port for your sound output. The 6522 timer 1 is capable of repeatedly toggling this pin at programmable intervals. This means that a pitch must be selected once and will remain until the device is reprogrammed. Before describing the programming of the device, we must first consider the hardware necessary to connect the port to an amplifier.

The output ports switch +5V or 0V to their loads. For use with a normal amplifier, this level must be lowered to suit the input of the amplifier used. A simple voltage divider will suffice. If you intend on using two channels, just double the resister value you would use for one channel and apply it to each. A circuit diagram will be given next month.

The software is even simpler. To set up the port as an oscillator, store $C0 in address $AX0B (X selects the 6522 to be used). Next, select the frequency by putting a 16 bit number in addresses $AX04 and $AX05 (low byte in $AX04 first). The higher the value used, the lower the frequency. A value of $0000 will generate a frequency of about 300 kHz and will inaudible. Next month I will publish a table of values for generating the musical scale.

There is more to a tune than just a number of notes. The waveshape determines the tonal quality of the sound. Tones produced by either of the above methods will sound 'bright'. Generating tones with a different quality requires a bit more effort. The old +5V/0V output is no longer sufficient and the port must be connected to a D/A converter in order to get full control of the waveform. A circuit diagram for such a device will be given next month.

The I/O port to which this circuitry must be connected has to be initialized as an output by storing $FF in $AX02 for port B or $AX03 for port A. Two locations below these are the corresponding output port addresses where the voltage level on the D/A must be written. The data can be fetched from a waveshape table and copied to the I/O port. The rate at which the data is copied determines the frequency of the tone. Harmonies can be produced purely in software by adding the data from two waveshapes together. Each table will have to be read at different rates to produce two different frequencies, but this should prove to be no problem. Care must be taken to ensure that the result of the addition is less than $FF or it will produce distortion due to the loss of the carry bit. This can be done by restricting the table values to +$1F or -$1F (which is equal to $E1). Before being applied to the I/O port, the data must be converted to a centre-zero format. ie. $80 in signed two's complement gives a result of $00, $00 gives $80 and $7F gives $FF with values in between adjusted in a similar fashion. This is done by toggling the most significant bit. The output from the D/A converter will have about a 2.5V bias.

NEXT MONTH - CIRCUIT DIAGRAMS plus ???

This has been the last of the series on the SYM. I hope it has been helpful in whatever you are doing. As promised, the musical tones table will be appearing next month.

If you have done something interesting with your SYM, write it up as an article and send it to me so it can be published. Others will be interested. Photocopies of magazine articles are no good as they are usually too long. If this is the case, extract the important information and write that up. As a kick off, our newest member, Paul Webber, will review the SCVT VDU published in EA (Sept. 1980) for those still without a terminal.

It's now up to you!

Brian Campbell

****

THE MEETING WAS KAOS

This is first report from your new roving reporter. There is not a lot to report at the present. The development 'committee' seems to have slowed down a bit and limiting themselves, from pure necessity, to a few major projects. Maybe they are looking for ideas??

David Anear demonstrated his new OSI HiRes colour board patched into his new RGB colour monitor via his self designed RGB driver mod. The display was great and not at all slow. He has also re-worked his Breakforth program to run in colour (plus a few other mods to the program).

Speaking of FORTH, come on fig-group, where are all your new screens?? While you're at it, don't forget the cassette users.

The continuing saga of GTBUG. Due to mechanical problems, Tony Durrant's now infamous GTBUG is still under wraps, but with a little luck, will be ready for time trials for next years Bathurst 5000. I think he is having memory trouble, (computer, not brain).

The 48K Boards also ran into a snag. This time it is the manufacturer, Phillips, in Sydney. They botched a few of the first run, then decided to close down their plant. There are a few boards available but with no silk screen component overlay. However, they are just as good but not as pretty. Contact David Anear in relation to these. More will be available soon, (not too long I hope).

The magazine library is now being run by Ron Kerry, so we should see this recent and popular service return to the club rooms next meeting.

The club is very quickly running out of helping hands. With about 400 members, and growing every week, it seems strange and unreasonable that only 6 or 7 have to do all the work. Where is your sense of responsibility, duty, loyalty and fairplay? Remember, the more people doing something, the less each has to do. So the next time volunteers are asked for, take a step forward. You will certainly be appreciated. This applies to club duties, as well as hardware and software help.

One last sour grape. What has happened to all the early arrivers? The number of computers available for the morning session has been dropping each month, making it difficult for the kids to get 'hands-on experience'. Now that we are actually teaching them programming, we need as many systems, at the sessions, as possible. Or at least as many as the low volts will allow us to run. We are asking for VOLUNTEERS!!!

Ron Cork

DEAR ~~ABBY~~ PAUL

Since there are quite a number of questions this month, I will arrange them under seperate headings.

PRINTERS

Q. What is the difference between parallel and serial printers?

A. Technically a parallel printer receives eight data bits, an earth and various 'handshaking' or control bits all down individual wires; a serial printer receives all the data bits, one at a time down one wire and an earth and a control bit (sometimes control bits) down other wires. As far as the 'front end user' is concerned, there is no difference.

Q. When I converted my Superboard to a C4P, I found that my printer no longer worked, why not?

A. The C1P's serial port is situated at $F000 and the C4P's serial port is at $FC00, there are two lines of code that you can add to your BEXEC* to make your "C4P" talk to the serial port at $F000 - refer to KAOS Vol.2, No.2, page 13.

Q. Sometimes my computer 'hangs' whilst outputting to my printer, why?

A. There are a number of answers to this problem, firstly I would suggest that you check your printer cable, the most likely cause is that the BUSY line from the printer or one of the other control lines is broken and is only making intermittent contact. Secondly, you could be suffering power problems with your printer, however this is far less likely than the first answer I gave.

DISK

Q. Is there anyway of fixing the recurring problem of the MPI 5" disk drives failing to eject the disk?

A. No, not really. The MPI drives have a poorly designed catch where one piece of spring steel catches against another, raising it and releasing a spring which ejects the disk - in theory. What happens is that the first piece of spring-steel becomes bent and fails to catch. You can try re-bending it, but the problem will soon re-appear.

Q. I recently read an article about loading and unloading the head on disk drives - should this be done?

A. If you have a MPI B51/52 then there is an easy mod to this drive - but I don't suggest that you do it. Rather, build a motor control circuit (refer KAOS Vol.2, No.8, page 14) and select the "Head load with motor on" option on the disk drive. Unloading the head greatly increases the life of the head and disk surfaces. Switching the motor will greatly increase its life.

Q. How often should disk drive heads be cleaned?

A. If you must use a head cleaning disk, I would suggest that you only occasionally use it - ie. every few months or so. If you start to get read/write errors then I would suggest that you clean the heads before suspecting a hardware fault.

GRAPHICS

Q. I still await the Hi-Res board, are there any good routines to use it?

A. There have been several Hi-Res boards around (prototypes only), however, none have gone into production and it is still uncertain whether any will. Ohio have lots of software available for their Hi-Res board which could be converted for use with any Hi-Res board. If you feel the urge to write your own software I suggest you purchase a book, one that I use, "Principals of Interactive Computer Graphics" by Newman and Sproull, published by McGraw - Hill.

If you have any questions suitable for inclusion on this page, write to DEAR ABBY c/o KAOS.

ARTICLES PLEASE???

by Garnett Znidaric

Talking with our valiant secretary after the last meeting about the lack of suitable articles and the problems associated with getting them to print, I started thinking about writing a short article about writing articles for the KAOS newsletter, which I feel, for the most part, surpasses all those magazines which always seem full of ads and irrelevent information. So here are a few prompts and comments for future articles.

When I joined KAOS I never once considered that maybe I might one day know enough about computers to submit anything which maybe of interest to other members. Well, this article in itself reinforces that comment, although I have from time to time been able to submit something of use to our 6502 community.

As for newsletter articles, they should be brief and to the point. As for topics, well, here are some articles that I would like to read.

1. An article on screen format design relating to setting the screen up for a game or business utility.
2. On the same vein, maybe an article on the disassembly of a game showing how each portion functions.
3. A semi detailed explanation of the USR(X) function.
4. A "comprehensive" guide to trouble shooting problems, using a multimeter, logic probe and CRO.
5. A listing of the chips commonly used and a quick explanation of how they work.
6. Software reviews such as WP6502 and the new CDMS system.
7. A discussion on understanding digital circuits.
8. Reviews on books that may be of use to either programmers or hackers.

In summary, we all have areas of knowledge and areas with large gaps. Using the spirit of club fellowship, I would like to glean my missing information from others, while I help fill in their gaps. So do us a favour and pull out that word processor and pound out an article. If need be, I'll even type it up to help Rosemary out.

WORDPROCESSING WITH COMP-DOS

As most people do when they get a new toy, I was eager to try out my new disk as soon as I got home with the 1.2 version of the renowned WP6502. After re-aquainting myself with my keyboard, with all the keys behaving as they should, I found a few cumbersome problems which I have since solved. Solved to my satisfaction at any rate.

Firstly I replaced the stock BEXEC* with the one on COMP-DOS as written by George except that I deleted the choice of DOS and as a final command asked for DISK!"DR. This then boots up with a directory listing of the files on the disk. An input query hangs the machine before it fires up the WP6502. This also allows you to get into the programme before the WP fires up so that you can play with it if need be.

The main point in changing the BEXEC*'s over was that with George's DOS, it is not necessary to create the file space for your document before it is written. (You can imagine the possible problems that could occur!!) Once again, thankyou George.

As for the WP itself. I couldn't let it go unchanged either, could I, so after I had the system running the way that I wanted it to as far as formats are concerned, I inserted a line between 100 and 105 saying GOTO 480 which is the direction after the changes have been made.

In making these changes I feel that this great 1.2 version is even more powerful and much easier to use for the un-initiated and great even for those who have been.

PS If you have gone C4 then you will probably have to open up the serial port as shown in the KAOS newsletter 2-2-13 at the bottom of the page.

Garnett Znidaric

MEMORY FILES AND HOW THEY ARE USED.

Have you ever wondered how Pico Dos sets the memory size when it wakes up or how OS65D3 loads and runs the program called BEXEC*. This article, I hope, will help you explore the method used.

When OS65D3 boots in it loads BASIC then jumps to BASIC and waits for an input. But as disk users know, instead of a keyboard input, it gets a command from a memory file which tells it to RUN"BEXEC*". On OS65D3 this command file is located at $2E1E. The following is program which loads track 1 and displays the command file

```
10 DISK!"CA 4A00=01,1":A=19998:REM $2E1E IN DECIMAL
20 FOR X=ATOA+16:PRINTCHR$(PEEK(X));:NEXT
```

If you wish to change the name of the first program loaded, all you need do is modify the command file to run the program you want. The following program will allow you to change the name of the first program loaded.

```
5DISK!"CA 4A00=01,1":A=20009
10INPUT"NEW NAME FOR BEXEC*";A$
15IFLEN(A$))6THENGOTO10
20FORX=1TO7-LEN(A$):A$=A$+CHR$(32):NEXT
25FORX=0TO5:POKEA+X,ASC(MID$(A$,X,1)):NEXT
30DISK!"SA 01,1=4A00/8"
35END
```

..................Please note there must be a program on the disk with the same name as that created in the above program. Also the first lines of this program should be as follows.

```
10 REM CHANGE INPUT AND OUTPUT TO ALLOW USER ACCESS
15 X=PEEK(10950):POKE8993,X:POKE8994,X
```

These lines reset the I/O to user input............................

To test the disk that you have changed RUN the following program
10DISK!"CA 4A00=01,1":20DISK!"MEM 4E1E,4E1E"
30DISK!"IO 10,02"   If the change is correct then the new program will load and run.

So far I have not explained how the indirect file operates, I will use the above program to explain to you line by line how it works.:
......Line 10. This operation brings the code on track 1 and places it in memory starting at location $4A00.
......Line 20. This line sets the starting address of the command.
......Line 30. This operation re-routes the input routine so that it takes the Ascii data from memory starting from the location setup by line 20. It will fill the input buffer until it find a carriage return character, it then passes this data to the operating system.

Next page please.

****

HIDDEN FILE ON PICO DOS.

The original Pico Dos code was written for Shuggart 35 track drives. If you have a 39 track drive there are 4 unused tracks that can be used as an extra file. This file is enabled by a single poke POKE 9656,9 into the operating system. This will allow the extra file (LOAD9) to be used. To make this modification permanent you will need to use the track zero copy utility to change location $25B8 to 09 and then save back onto track zero.

****************** ***

David Alexander of   is offering a printer ribbon re-inking service. He points out that usually the problem with faint print is not a worn out ribbon but merely a lack of ink. The re-inking charge is $6.00 per cartridge, including return postage.

****

Integrated Business Services of 8 Drake St, Springwood, QLD 4127 offer a box of Nashua 5 1/4" S/S D/D disks, in a case, for $45.00 inc. tax. I.S.B. also sell peripheral equipment, including Epson Printers.

Now we can use what we have learned to create a program which puts a command file as entered by the user into memory and runs it.

```
5POKE133,115:POKE132,255:CLEAR:REM FOR 32K
10POKE2972,13:POKE2976,13:POKE2888,0:POKE8722,00
15DISK!"MEM 7400,7400"
20PRINT"WHAT IS YOUR LIST OF COMMANDS"
25INPUTA$:IFA$=""THENGOTO35
30PRINT#5,A$:GOTO25
35PRINT#5,"DISK!"+CHR$(34)+"IO 02,02"
40DISK!"IO 10,02"
```

......Line 5. First we need to set aside some RAM for the file. Location 133 is used by BASIC to tell it what the highest memory location it can use.
......Line 10. Because commas and colons are used to drive BASIC and DOS we have to disable them also we need to know when we have finished with the file.
......Line 15. The operating system needs to be set up in the reserved memory, this gives us the use of 1K for our command file.
......Line 20. Prompt for user.
......Line 25. Data is inputed and a check made to see if user is finished.
......Line 30. Basic device #5 stores the data in print statement into memory relative to the MEM command. Each time a character is outputed the MEM address is automatically incremented.
......Line 35. The last instruction in the file should return control to the user. Device #5 is used to store print statement into MEMory.
......Line 40. This is used to transfer control to the memory file and the entered commands should be seen on the screen as they are processed.

Any valid BASIC command can be used. Pressing RETURN only causes a jump to the command file.

Jeff Rae

****

65U DOS RAMBLINGS

Continued from last month

In last month's article on lines 5, 11 and 27, read 65U as 65D - and add the words 'are added to 65U' to the end of the article.

Two of the most useful and powerful BASIC statements provided are the keywords "INDEX" and FIND$. Before going into detail about these words, let me explain some of the concepts of DATA handling.

For example let us decide that we wish to CREATE a list of names, address', phone numbers and some comments on say - 50 people.

Once we have put all this information together, it will become a FILE of data. If we store this somewhere, either in a filing cabinet or on a mass storage device such as a floppy disk, we need to label or name it for later use. After all, we are sure to have other files. So! We now have a file of 50 names, address and other particulars. Each group of particulars comprises of a number of FIELDS of data. A field may be the Name as field 1. The address as Field 2, the phone number as Field 3 and so on. In fact, typically, a list such as this would have many more Fields arranged possibly like this.

Field 1	=SURNAME	Field 2	=INITIALS
Field 3	=STREET	Field 4	=TOWN
Field 5	=STATE	Field 6	=PHONE
Field 7	=COMMENTS		

One complete set of FIELDS (7) is called a RECORD and in our example there were 50 RECORDS in our FILE.

Each one of our Fields must be assigned a VARIABLE name so that the Basic language can operate on it, for example STREET=ST$(1TO50) TOWN=TN$(1TO50) and so on.

The next thing we need to decide is how many characters each of our FIELDS will need to be. Obviously, it would be wasteful of disk or memory space to allocate 25 characters for each field so let us FORMAT our needs. We might decide that for SURNAMES, 25 characters will be enough. INITIALS we can do with 3, STREETS might need another 25, TOWNS we will need 20, STATES we will abbreviate and use only 3 characters, for the PHONE we will never need more than 9 digits in Australia. Less if local numbers only. COMMENTS might take up 50 characters. Thus, the number of characters maximum for each of our 50 RECORDS will be 25+3+25+20+3+9+50, which makes each RECORD a maximum of 135 characters long. For 50 Records or our Complete FILE, we need to set aside storage of 135 * 50 which is very close to 7750.

INDEX(X)=Y With 65U, I mentioned that we can have up to 8 channels opened where a channel contains a block of Data which at this time we don't know what it is, but we do know its' FORMAT. X is the number of the channel that we wish to work on and Y is = to the INDEX of the Field that we wish to use. First of all let us decide that the first 10 character locations are going to store a special number. This number could be (and normally is) the number of the total number of characters stored in the file at that time. This saves us having to look through the whole 7750 characters each time we want something if we really only had 3 or 4 records stored.

David Tasker

Continued next month

****

FOR SALE

OSI C4P, ex. cond., power supply, documentation & software. BARGAIN $450.00
C8P with 48K, D/S, D/D. Sanyo Green Screen, plus $2000 worth of software, $5000
Contact Scott Parker

****

APPLE 48K Board & Power Supply, $480.00, KEYBOARD $220.00, Parallel Interface & Cable $100.00, Disk Controller Card $100.00.
All seem working, phone Alan on

****

SYM-1 and KTM-2. Dual +5V6A Power Supply (permanent connection to both modules) RF Modulator (CH 0) connected. All manuals and handbooks, plus other micro-computer books (games, applications, "A.P.C.") Cost $450.00 ono
Contact Steve

****

SUPERBOARD II in Metal Case, complete with power supply, 8K RAM, DABUG III, 48 character. Included - Cassette player with leads, game software cassettes, manuals & circuits. * $360.00 * ono
Ring Peter McLennen, after hours

****

Ex PMG TELETYPE in working condition. Complete with 110V-250V transformer and ASC11-BADOUT conversion program in EPROM and 20m A LOOP. $75.00
Contact Tony Van Bergen

****

FOR SALE: A bare Disk Controller Board for the Tasker Buss. Price is $15.00
Contact Warren Read,

****

C1-P in case fitted with Geoff Cohen's C1 to C4 video board modified for use as C1-P or C4-P switchable. 610 expansion board, 32K RAM ,floppy disk interface switchable between 5.25 or 8 inch drives. Two Qume 5.25 inch double sided double density drives in case with power supplies. Supplied with 30 diskettes, cassette tapes with assorted software.
Worth over $1800 sell $1400 ONO. Contact Jeff Rae